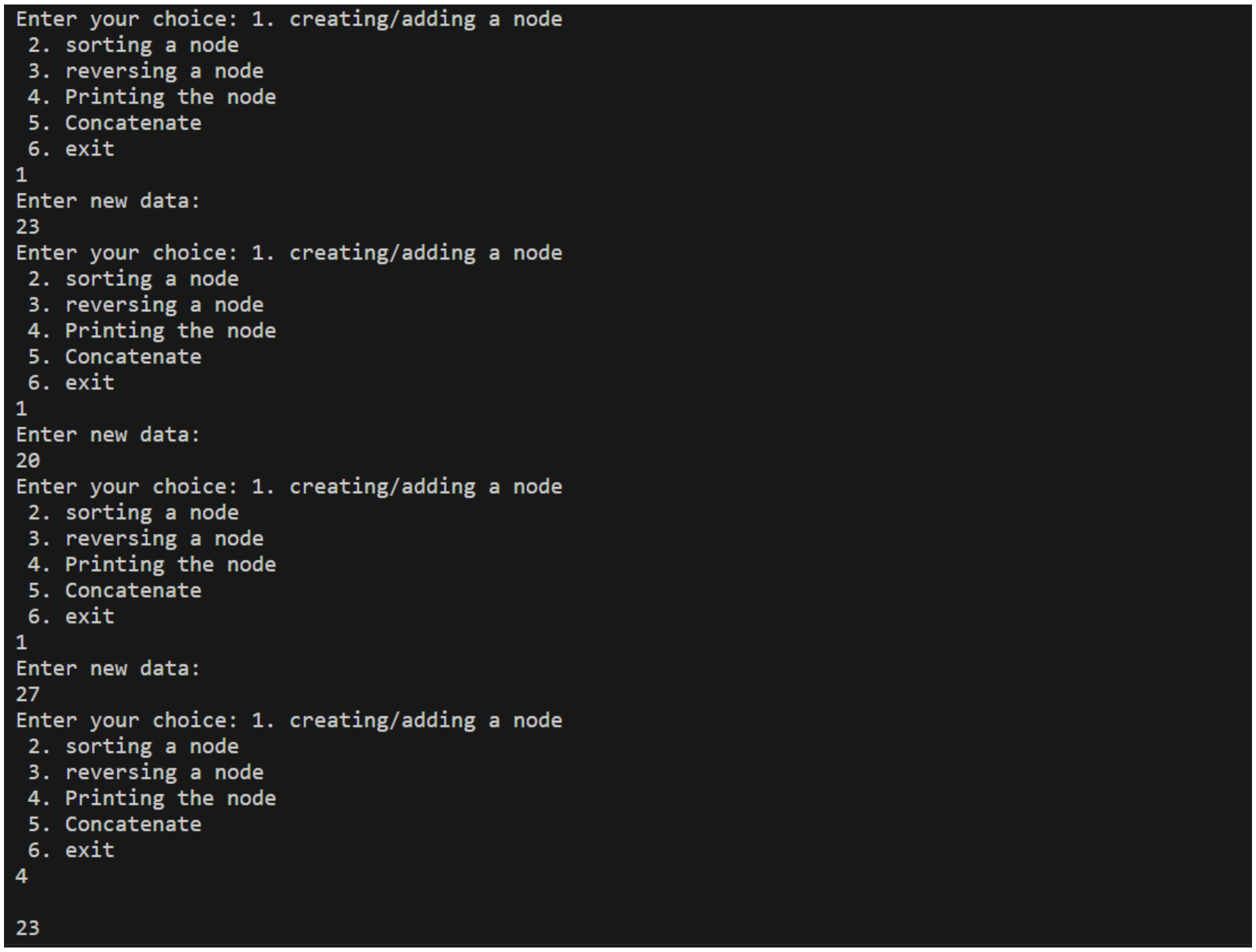
1BM22CS260-SHLOK IYER SECTION:CE SEM:3

6a) WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked

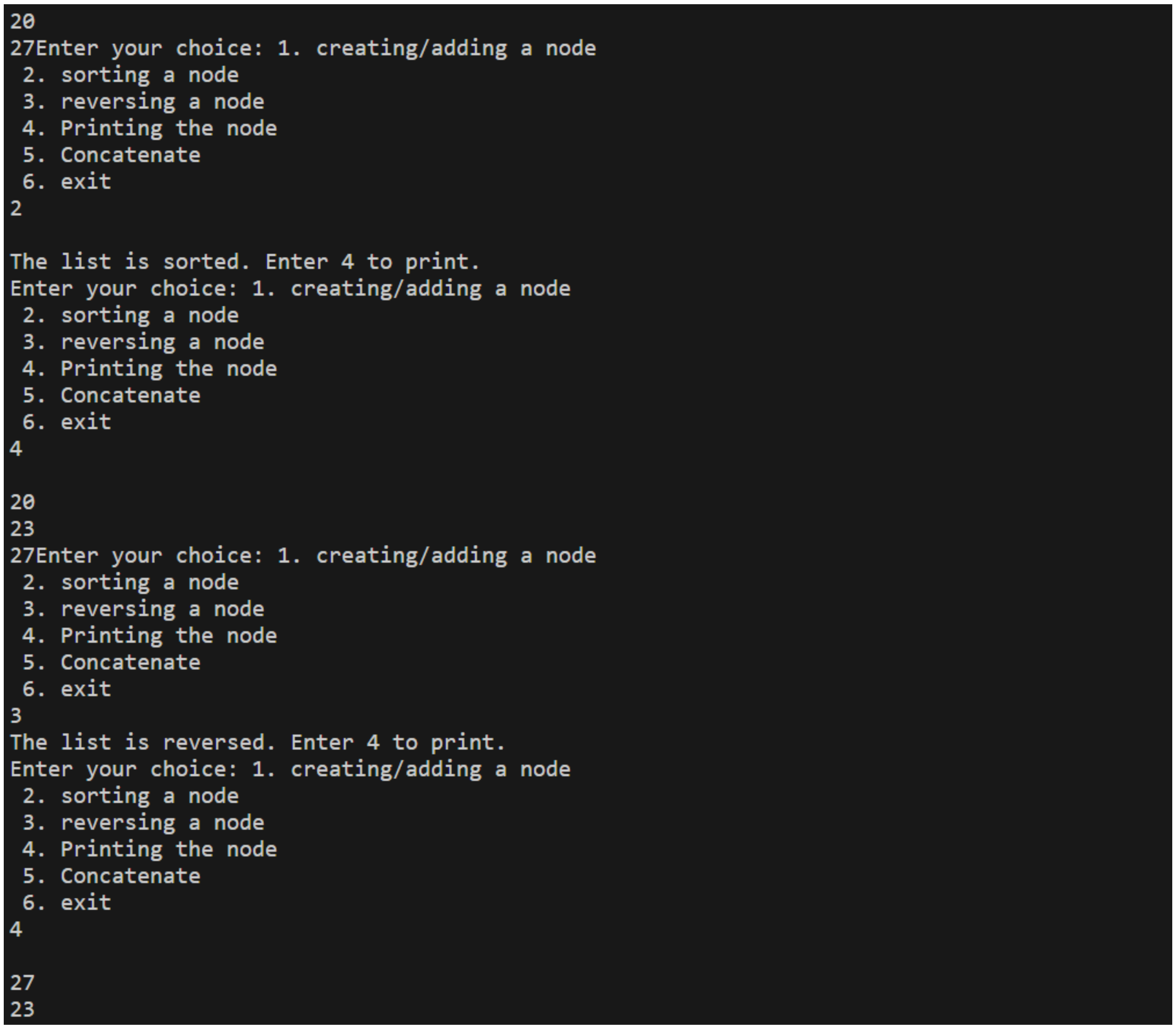
list, Concatenation of two linked lists.

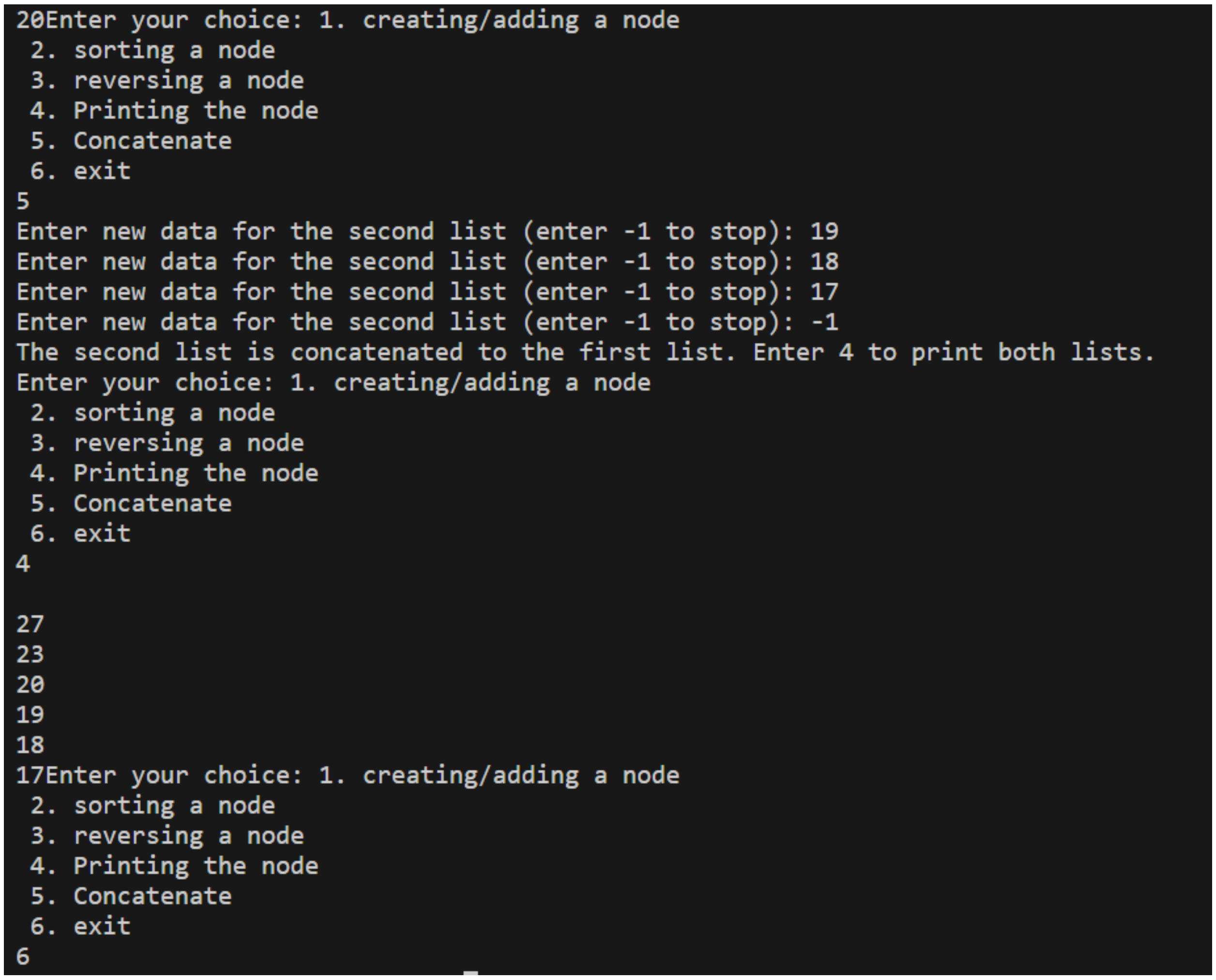
```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int data;
    struct node *next;
};

struct node *head;  // Declare head globally

void reverseLL(struct node **head_ref)
{
    struct node *prev = NULL;
    struct node *next = NULL;
    struct node *current = *head_ref;
    while (current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
}

void PushNode(struct node **head_ref, int new_data)
{
    struct node *new_node = (struct node *)malloc(sizeof(struct node));
    struct node *last = *head_ref;
    new_node->data = new_data;
    new_node->next = NULL;
    if (*head_ref == NULL)
    {
        *head_ref = new_node;
        return;
    }
    while (last->next != NULL)
    {
        last = last->next;
    }
    last->next = new_node;
}

void printLL(struct node *head)
{
    struct node *current;
    current = head;
    while (current != NULL)
    {
        printf("\n%d", current->data);
        current = current->next;
    }
}

void sortLL(struct node *head)
{
    struct node *current = head, *index = NULL;
    int temp;
    if (head == NULL)
    {
        printf("Cannot reverse");
    }
    else
    {
        while (current != NULL)
        {
            index = current->next;
            while (index != NULL)
            {
                if (current->data > index->data)
                {
                    temp = current->data;
                    current->data = index->data;
                    index->data = temp;
                }
                index = index->next;
            }
            current = current->next;
        }
    }
}

void ConcatLL(struct node **head1_ref, struct node *head2)
{
    struct node *last = *head1_ref;
    while (last->next != NULL)
        last = last->next;
    last->next = head2;
}

int main()
{
    head = NULL;  // Initialize head to NULL
    struct node *new_list = NULL;  // Declare new_list globally

    while (1)
    {
        int ch;
        printf("Enter your choice: 1. creating/adding a node\n 2. sorting a node\n 3. reversing a
node\n 4. Printing the node\n 5. Concatenate\n 6. exit\n");
        scanf("%d", &ch);

        switch (ch)
        {

        case 1:
        {
            int new_data;
            printf("Enter new data:\n");
            scanf("%d", &new_data);
            PushNode(&head, new_data);
            break;
        }
        case 2:
        {
            sortLL(head);
            printf("\nThe list is sorted. Enter 4 to print.\n");
            break;
        }
        case 3:
        {
            reverseLL(&head);
            printf("The list is reversed. Enter 4 to print.\n");
            break;
        }
        case 4:
        {
            printLL(head);
            break;
        }
        case 5:
        {
            int new_data;
            while (1)
            {
                printf("Enter new data for the second list (enter -1 to stop): ");
                scanf("%d", &new_data);
                if (new_data == -1)
                    break;
                PushNode(&new_list, new_data);
            }
            ConcatLL(&head, new_list);
            printf("The second list is concatenated to the first list. Enter 4 to print both lists.\n");
            break;
        }
        case 6:
        {
            exit(0);
        }
        }
    }

    return 0;
}
```

OUTPUT:



```
Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
1
Enter new data:
23
Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
1
Enter new data:
20
Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
1
Enter new data:
27
Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
4

23
```


```
20
27Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
2

The list is sorted. Enter 4 to print.
Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
4

20
23
27Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
3
The list is reversed. Enter 4 to print.
Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
4

27
23
```



```
20Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
5
Enter new data for the second list (enter -1 to stop): 19
Enter new data for the second list (enter -1 to stop): 18
Enter new data for the second list (enter -1 to stop): 17
Enter new data for the second list (enter -1 to stop): -1
The second list is concatenated to the first list. Enter 4 to print both lists.
Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
4

27
23
20
19
18
17Enter your choice: 1. creating/adding a node
 2. sorting a node
 3. reversing a node
 4. Printing the node
 5. Concatenate
 6. exit
6
```